WE CLAIM:

1. Apparatus for processing data comprising:

processing logic operable to perform data processing operations; and

an instruction decoder operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions; wherein

said instruction decoder is responsive to a memory access instruction:

(i) to compare a base register value stored within a base register specified by a base register field of said memory access instruction with a predetermined null value; and

(ii) if said base register value matches said predetermined value, then to branch to execution of a null value exception handler.

2. Apparatus as claimed in claim 1, wherein in response to said memory access instruction a return address is stored pointing a memory location storing a program instruction to be executed upon a return from said null value exception handler.

3. Apparatus as claimed in any one of claim 1 and 2, wherein said null value exception handler is located at a memory address pointed to by a value stored within a programmable configuration register.

4. Apparatus as claimed in claim 3, wherein said programmable configuration register is a coprocessor configuration register.

5. Apparatus as claimed in any one of claims 3 and 4, wherein said branch is made to an instruction stored at a memory address given by said value stored within said programmable configuration register subject to a fixed offset.

6. Apparatus as claimed in any one of the preceding claims, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to emulation

of a non-native program instruction that is not directly decodable by said instruction decoder attempting to make a memory access using a null value.

7. Apparatus as claimed in any one of the preceding claims, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to an error in operation of a virtual machine computer program operable to translate non-native program instructions that are not directly decodable by said instruction decoder into native program instructions that are directly decodable by said instruction decoder.

8. Apparatus as claimed in any one of claims 6 and 7, wherein said non-native program instructions are machine independent program instructions.

9. Apparatus as claimed in claim 8, wherein said machine independent program instructions are one of:
Java bytecodes;
MSIL bytecodes;
CIL bytecodes; and
.NET bytecodes.

10. Apparatus as claimed in any one of claims 6 and 7, wherein said non-native instructions are native program instructions of a different apparatus for processing data.

11. Apparatus as claimed in claim 10, wherein said processing logic and said instruction decoder are part of a RISC processor and said non-native instructions are native instructions of a CISC processor.

12. Apparatus as claimed in claim 3 and any one of claims 2 and 4 to 11, wherein said value stored within said programmable configuration register is a start address of said null value exception handler.

13. Apparatus as claimed in claim 3 and any one of claims 2 and 4 to 11, wherein said value stored within said programmable configuration register is an address of a jump instruction operable to jump execution to a start address of said null value exception handler.

14. Apparatus as claimed in any one of the preceding claims, wherein said memory access instruction is a load instruction operable to load into a destination register specified by a destination register field within said load instruction a load value dependent upon a value read from a memory location specified by said base register value.

15. Apparatus as claimed in any one of claims 1 to 13, wherein said memory access instruction is a store instruction operable to store into a memory location specified by said base register value a store value dependent upon source value stored within a source register specified by a source register field within said store instruction.

16. A method of processing data with an apparatus for processing data having processing logic operable to perform data processing operations and an instruction decoder operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions, said method comprising the steps of:

in response to said memory access instruction decoded by said instruction decoder controlling said processing logic:

(i) to compare a base register value stored within a base register specified by a base register field of said memory access instruction with a predetermined null value; and

(ii) if said base register value matches said predetermined value, then to branch to execution of a null value exception handler.

17. A method as claimed in claim 16, wherein in response to said memory access instruction a return address is stored pointing a memory location storing a program instruction to be executed upon a return from said null value exception handler.

18. A method as claimed in any one of claim 16 and 17, wherein said null value exception handler is located at a memory address pointed to by a value stored within a programmable configuration register.

19. A method as claimed in claim 18, wherein said programmable configuration register is a coprocessor configuration register.

20. A method as claimed in any one of claims 18 and 19, wherein said branch is made to an instruction stored at a memory address given by said value stored within said programmable configuration register subject to a fixed offset.

21. A method as claimed in any one of claims 16 to 20, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to emulation of a non-native program instruction that is not directly decodable by said instruction decoder attempting to make a memory access using a null value.

22. A method as claimed in any one of claims 16 to 21, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to an error in operation of a virtual machine computer program operable to translate non-native program instructions that are not directly decodable by said instruction decoder into native program instructions that are directly decodable by said instruction decoder.

23. A method as claimed in any one of claims 21 and 22, wherein said non-native program instructions are machine independent program instructions.

24. A method as claimed in claim 23, wherein said machine independent program instructions are one of:

Java bytecodes;

MSIL bytecodes;

CIL bytecodes; and

.NET bytecodes.

25. A method as claimed in any one of claims 21 and 22, wherein said non-native instructions are native program instructions of a different apparatus for processing data.

26. A method as claimed in claim 25, wherein said processing logic and said instruction decoder are part of a RISC processor and said non-native instructions are native instructions of a CISC processor.

27. A method as claimed in claim 18 and any one of claims 17 and 19 to 26, wherein said value stored within said programmable configuration register is a start address of said null value exception handler.

28. A method as claimed in claim 18 and any one of claims 17 and 19 to 26, wherein said value stored within said programmable configuration register is an address of a jump instruction operable to jump execution to a start address of said null value exception handler.

29. A method as claimed in any one of claims 16 to 28, wherein said memory access instruction is a load instruction operable to load into a destination register specified by a destination register field within said load instruction a load value dependent upon a value read from a memory location specified by said base register value.

30. A method as claimed in any one of claims 16 to 28, wherein said memory access instruction is a store instruction operable to store into a memory location specified by said base register value a store value dependent upon source value stored within a source register specified by a source register field within said store instruction.

31. A computer program product including a computer program operable to control an apparatus for processing data having processing logic operable to perform data processing operations and an instruction decoder operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions, said computer program comprising:

a memory access instruction decodable by said instruction decoder to control said processing logic:

(i) to compare a base register value stored within a base register specified by a base register field of said memory access instruction with a predetermined null value; and

(ii) if said base register value matches said predetermined value, then to branch to execution of a null value exception handler.

32. A computer program product as claimed in claim 31, wherein in response to said memory access instruction a return address is stored pointing a memory location storing a program instruction to be executed upon a return from said null value exception handler.

33. A computer program product as claimed in any one of claim 31 and 32, wherein said null value exception handler is located at a memory address pointed to by a value stored within a programmable configuration register.

34. A computer program product as claimed in claim 33, wherein said programmable configuration register is a coprocessor configuration register.

35. A computer program product as claimed in any one of claims 33 and 34, wherein said branch is made to an instruction stored at a memory address given by said value stored within said programmable configuration register subject to a fixed offset.

36. A computer program product as claimed in any one of claims 31 to 35, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to emulation of a non-native program instruction that is not directly decodable by said instruction decoder attempting to make a memory access using a null value.

37. A computer program product as claimed in any one of claims 31 to 36, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to an error in operation of a virtual machine computer program operable to translate non-native program instructions that are not directly decodable by said instruction decoder into native program instructions that are directly decodable by said instruction decoder.

38. A computer program product as claimed in any one of claims 36 and 37, wherein said non-native program instructions are machine independent program instructions.

39. A computer program product as claimed in claim 38, wherein said machine independent program instructions are one of:
Java bytecodes;
MSIL bytecodes;
CIL bytecodes; and
.NET bytecodes.

40. A computer program product as claimed in any one of claims 36 and 37, wherein said non-native instructions are native program instructions of a different apparatus for processing data.

41. A computer program product as claimed in claim 40, wherein said processing logic and said instruction decoder are part of a RISC processor and said non-native instructions are native instructions of a CISC processor.

42. A computer program product as claimed in claim 33 and any one of claims 32 and 34 to 41, wherein said value stored within said programmable configuration register is a start address of said null value exception handler.

43. A computer program product as claimed in claim 33 and any one of claims 32 and 34 to 41, wherein said value stored within said programmable configuration register is an address of a jump instruction operable to jump execution to a start address of said null value exception handler.

44. A computer program product as claimed in any one of claims 31 to 43, wherein said memory access instruction is a load instruction operable to load into a destination register specified by a destination register field within said load instruction a load value dependent upon a value read from a memory location specified by said base register value.

45. A computer program product as claimed in any one of claims 31 to 43, wherein said memory access instruction is a store instruction operable to store into a memory location specified by said base register value a store value dependent upon source value stored within a source register specified by a source register field within said store instruction.

46. A computer program product including a computer program operable to translate non-native program instructions to form native program instructions directly decodable by an apparatus for processing data having processing logic operable to perform data processing operations and an instruction decoder operable to decode program instructions to control said processing logic to perform data processing operations specified by said program instructions, said native program instructions comprising:

a memory access instruction decodable by said instruction decoder to control said processing logic:

(i) to compared a base register value stored within a base register specified by a base register field of said memory access instruction with a predetermined null value; and

(ii) if said base register value matches said predetermined value, then to branch to execution of a null value exception handler.

47. A computer program product as claimed in claim 46, wherein in response to said memory access instruction a return address is stored pointing a memory location storing a program instruction to be executed upon a return from said null value exception handler.

48. A computer program product as claimed in any one of claim 46 and 47, wherein said null value exception handler is located at a memory address pointed to by a value stored within a programmable configuration register.

49. A computer program product as claimed in claim 48, wherein said programmable configuration register is a coprocessor configuration register.

50. A computer program product as claimed in any one of claims 48 and 49, wherein said branch is made to an instruction stored at a memory address given by said value stored within said programmable configuration register subject to a fixed offset.

51. A computer program product as claimed in any one of claims 46 to 50, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value corresponds to emulation of a non-native program instruction that is not directly decodable by said instruction decoder attempting to make a memory access using a null value.

52. A computer program product as claimed in any one of claims 46 to 51, wherein said null value exception handler is operable to determine if said memory access instruction attempting to access a location corresponding to a null value

corresponds to an error in operation of a virtual machine computer program operable to translate non-native program instructions that are not directly decodable by said instruction decoder into native program instructions that are directly decodable by said instruction decoder.

53. A computer program product as claimed in any one of claims 51 and 52, wherein said non-native program instructions are machine independent program instructions.

54. A computer program product as claimed in claim 53, wherein said machine independent program instructions are one of:
Java bytecodes;
MSIL bytecodes;
CIL bytecodes; and
.NET bytecodes.

55. A computer program product as claimed in any one of claims 51 and 52, wherein said non-native instructions are native program instructions of a different apparatus for processing data.

56. A computer program product as claimed in claim 55, wherein said processing logic and said instruction decoder are part of a RISC processor and said non-native instructions are native instructions of a CISC processor.

57. A computer program product as claimed in claim 48 and any one of claims 47 and 49 to 56, wherein said value stored within said programmable configuration register is a start address of said null value exception handler.

58. A computer program product as claimed in claim 48 and any one of claims 47 and 49 to 56, wherein said value stored within said programmable configuration register is an address of a jump instruction operable to jump execution to a start address of said null value exception handler.

59. A computer program product as claimed in any one of claims 46 to 58, wherein said memory access instruction is a load instruction operable to load into a destination register specified by a destination register field within said load instruction a load value dependent upon a value read from a memory location specified by said base register value.

60. A computer program product as claimed in any one of claims 46 to 58, wherein said memory access instruction is a store instruction operable to store into a memory location specified by said base register value a store value dependent upon source value stored within a source register specified by a source register field within said store instruction.

61. Apparatus for processing data substantially as hereinbefore described with reference to the accompanying drawings.

62. A method of processing data substantially as hereinbefore described with reference to the accompanying drawings.

63. A computer program product substantially as hereinbefore described with reference to the accompanying drawings.